Modelling Enterprise Architectures: An Approach Based on Linking Metaphors and Ontologies

Gerald R. Khoury
Institute for Information and Communication Technologies
University of Technology Sydney
PO Box 123 Broadway NSW 2007 Australia
grkhoury@it.uts.edu.au

Simeon J. Simoff
Institute for Information and Communication Technologies
University of Technology Sydney
PO Box 123 Broadway NSW 2007 Australia
simeon@it.uts.edu.au

John Debenham
Institute for Information and Communication Technologies
University of Technology Sydney
PO Box 123 Broadway NSW 2007 Australia
John.Debenham@uts.edu.au

Abstract

This paper describes a new approach to modelling enterprise architectures. By viewing enterprise models as metaphors and then linking these metaphors, in a structured way, to ontologies, we propose that unified languages can be developed to describe a wide variety of enterprise structures. This overcomes a serious shortcoming with contemporary approaches to enterprise modelling. This approach is operationalised and applied to develop a new unified modelling language that demonstrates the application of this methodology.[·]

Keywords: Enterprise Architecture, Modelling, Metaphor, Ontology

1 Introduction

Enterprise architectures (EAs) are growing in importance as tools for managing change within highly dynamic and competitive business environments. As the rate of technological change increases, and as the information environment becomes more complex, more sophisticated methods are needed to manage that change effectively. Enterprise architectures help to manage this change and overcome the problems of building isolated IT solutions that fail to support the enterprise's goals and objectives.

We define Enterprise architecture as *a holistic set of models that represent an enterprise's information systems in order to manage change*.

Enterprise architecture projects involve professionals with different backgrounds, who build a *shared understanding* of the IT solution structures. As a result such projects will benefit significantly from a unified modelling language that can be used to model all aspects of an enterprise (Noran, 2003). Instead, current practices use a wide variety of disparate modelling languages to create comprehensive EA models. This leads to several problems (Khoury and Simoff, 2005): (i) EA modellers are required to have expertise in a number of diverse modelling languages, which places a high cognitive load upon them; (ii) the use of disparate languages to represent a single architecture leads to *inconsistent semantics and weak ontologies* and, consequently, incoherent and inconsistent EA models; and (iii) due to their complexity, the resultant models are beyond the comprehension of those whom are not modelling experts, yet are interested in EA. This includes ICT and business *management*: perhaps the most important audience of enterprise architectures.

This paper addresses the problems discussed above. We propose a methodology for developing *unified* EA languages.[1] The practical application of the methodology is then illustrated through the development of an instance of a unified EA modelling language.

The proposed approach builds on complementary elements from two areas of research: our theoretical foundation for viewing models as metaphor, and Gruber's (Gruber, 1993) principles of designing ontologies for knowledge sharing. This perspective allows us to develop highly abstract metaphors that have particular efficacy for the modelling of a wide variety of organisational structures. As an explicit specification of a

[1] We describe an EA language as 'unified' if it can be used to model a range of diverse ICT domains.

conceptualization, “a set of categories of objects or ideas in the world, along with certain relationships among them” (Hirst, 2003), ontologies and the principles of their design are used in our approach for specialising specifications of the selected metaphors. The entire process is operationalised to provide a reusable methodology for the development of unified EA modelling languages.

In the following section, we describe this unique approach to the development of an integrated modelling language.

2 Enterprise Models and Metaphors

The traditional approach to solving the problem of developing unified EA models has been to attempt to develop higher-level abstractions from low-level, domain specific notations, by trying to find some way to combine and resolve the distinct notations. This approach has not been encouraging, and leads to the conclusion that "It seems very unlikely that we can develop a really general architecture framework that will simultaneously be formalizable.” (Maier and Rechtin, 2000)

Instead of using this ‘bottom-up’ approach, we take a ‘top-down’ approach to the problem. The first step in this process is to investigate high-level conceptual metaphors that can be used to describe various enterprise structures.

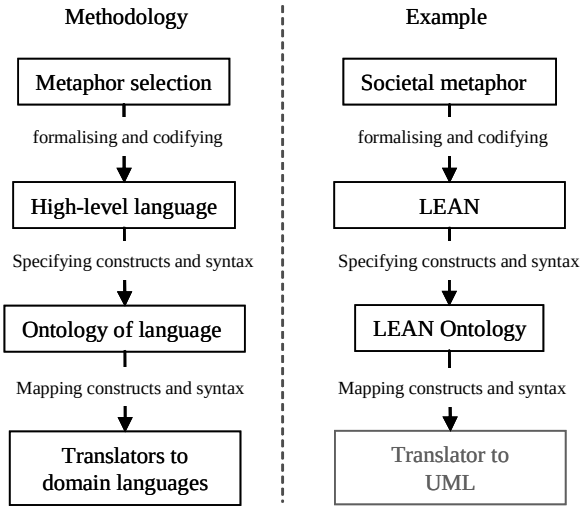

Fig. 1: A model of the metaphor-based approach to EA modelling

Much research has been focussed on the area of metaphor and its application to information technology. While the aim of this paper is not to provide a detailed review of the literature in this area, we present some observations that serve as a basis for developing an ontology suitable for EA modelling. Firstly, we assert that enterprise models do not represent reality: they create a *perception* of reality. Enterprise models are metaphors that actively shape the audience’s understanding of the organisation through the choice of complex underlying systems of assumptions and common understandings. This vision constitutes the basis of our approach towards building tools for the conceptual modelling of EA’s, as illustrated in Figure 1. The approach requires commitment to a particular metaphor, which then defines the components of the modelling language. Further, we present the details of proposed technique through an example, as also shown in Figure 1. The work on translators to domain languages is beyond the scope of this paper.

The following section illustrates how this view of enterprise modelling can lead us to new insights.

2.1 Metaphor Selection

As we have seen, any enterprise model that we develop will shape our understanding of the system. We note also, that metaphors form a dynamic type hierarchy (DTH) (Way, 1991). Since organisational models are metaphors, then they can also be viewed as a DTH[2]. This means that the development of an enterprise model necessitates the choice of some metaphor, and this choice has the following ramifications:

1. The metaphor influences our perception of the enterprise.
2. The model, if it is accurate, shares the common attributes of its subtypes.

The first criterion implies that we should be careful to choose a meritorious enterprise modelling metaphor. This may be assessed, for instance, by how well it aligns with corporate strategy, or perhaps it’s flexibility and applicability to a wide range of enterprises.

The second criterion implies that if we raise the level of abstraction of the enterprise metaphor, then we increase the number of sub-types, or system structures, that can be described by that metaphor. In order to develop a model that can be used to describe *any* enterprise, we need to create a single model that can describe the common features of all enterprise sub-structures, or systems. We can achieve this by creating a model that has a higher level of abstraction than the concept of ‘enterprise’ itself.

There are many candidates for this role. For example, the metaphor ‘the learning organisation’ is a highly abstract metaphor that can easily be applied to a wide range of organisations: hence its popularity. Another example is ‘an enterprise is a society’. We take this example and show how the choice of an appropriately abstract metaphor can be developed into a formal enterprise modelling language.

2.2 A Societal Metaphor for Enterprise Modelling

What does a model of society look like? There could be a range of answers to this question, but one well established model is that developed by Giddens (Giddens, 1984). According to Giddens' Theory of Structuration, there is interdependency between humans (actors) and societal

[2] The DTH is a generalisation hierarchy where the entities are categorised based on the common attributes. In a generalisation hierarchy, the higher-level class (supertype) shares the common attributes of the lower level class (subtype), while subtypes inherit all the properties of its supertype.

structures (resources and rules) that is manifest through specific actions. An actor is an individual who can exert power in order to produce an effect. Resources are "structured properties of social systems, drawn upon and reproduced by knowledgeable agents in the course of interaction." Rules refer to the sanctioned modes of conduct, and an action is an activity that is performed.

Note: if a different metaphor were chosen, then it would simply be a case of identifying the basic structures into which the metaphor can be composed, either by consulting the literature or by working from first principles. For example, Khoury & Simoff carried out this process using the concept of game-play as a metaphor source (Khoury and Simoff, 2003).

By formalising and codifying this conceptual metaphor, the theory of structuration provides the foundation for a modelling language that can be used to describe the structures contained within a wide variety of organisations. This is the subject of the next section of this paper.

4 LEAN Metamodel

We have shown how a highly conceptual metaphor can be used to provide the basis for an enterprise model that can describe a wide variety of enterprises. We then showed how a metaphor such as 'an enterprise is a society' can be structured; in this case, according to existing anthropological theory. In the next step, we take the societal metaphor and, based on our understanding of societal structures, construct a modelling language: the Lightweight Enterprise Architecture Notation (LEAN).

A metamodel defines the constructs and rules (or semantics) that are used to build models. It is simply the creation of a model at a higher level of abstraction than the thing being modelled (Henderson-Sellers and Bulthuis, 1998). That is, any given model is just an instance of the metamodel.

We defined four concepts, in the previous section, which are to be used as the foundation for LEAN. It is generally accepted that "… human understanding is improved by visual representations." (Polovina, 1993) There are clear benefits in being able to visualise an EA and graphical languages have a role in supporting visualisation. Given that all graphs consist of nodes connected by arcs, there are a wide variety of ways in which we could develop these four societal concepts into an ontology. However, taking into account our everyday understanding of the actual concepts to which we are referring, the most logical approaches can be summarised as follows:

1. Some of the structural concepts are represented as relationships, while others are represented as nodes.
2. All of the structural concepts are represented as nodes and a separate, but predefined, set of relationships connects them.
3. All of the structural concepts are nodes and a separate set of relationships connects them, but these relationships are *not* predefined within LEAN.

An example of the first case is the use of the concept of Rule to define the relationships between Agents, Actions and Resources. For example, 'an Agent performs an Action according to some Rule', or 'an Action produces or consumes a Resource according to some Rule'.

As an example of the second case we could define a fixed number of relationships to be used in the model and four node types to represent each of the societal concepts. For instance, we might define certain hierarchical relationships such as component-subcomponent and type-subtype relationships as being the only types of relationships permitted. All models would then have to be built connecting the Action, Agent, Resource and Rule nodes using only the pre-defined relationship set.

The third case is similar to the second, except that the relationships are not predefined. The enterprise can develop their own relationships on an ad-hoc basis according to their needs. This would allow the enterprise to take into account the nature of the information that they want to express and the context within which they are expressing it. This option clearly provides for the greatest expressive power. However, it does this at the expense of standardisation and, if the relationships are not well defined, formality.

As option 3 has the greatest expressive potential, we will use this as the basis for the development of a LEAN language syntax.

5 LEAN Syntax

We define the LEAN syntax as follows. The LEAN syntax is a graph notation composed of nodes and arcs. LEAN graphs contain one or more nodes, connected by zero or more arcs. An arc is connected to exactly two nodes, with one node attached to each end of the arc. A node is connected to zero or more arcs. However, each pair of nodes may only be connected by a single arc.

Thus, LEAN models may be connected (where there is a path between every pair of nodes in the graph) or disconnected graphs.

5.1 LEAN Nodes

In compliance with the definition of the societal metaphor in Section 2.2, LEAN contains four node types: (i) Agent; (ii) Action; (iii) Rule; and (iv) Resource. These are termed 'universal' types, since 'non-universal' types can also be represented as nodes. Non-universal types are subtypes of the universal types.

The LEAN nodes are defined as follows:

- An Agent is an entity that can exert power in order to produce an effect. In relation to IT systems, the immediate effect is the exchange of information. For example,

Agents may be people, roles, organisations, communities, nation-states or systems.

- A Resource is a structured property of the modelled system that can be consumed or produced by one or more Agents. For example, Resources may be raw materials, systems, documents, images, services or agents (in the case where agents signify constraints on the system).
- A Rule defines a sanctioned mode of conduct. For example, Rules can be used to represent physical constraints, logical constraints, legal and regulatory compliance. Rules can also be used to represent standards and guidelines.
- An Action is an activity that is performed. Actions equate to the capabilities that Agents possess. For example, an Action can involve the addition, modification, deletion, identification or selection of data, information or systems.

5.2 LEAN Arcs

A LEAN arc connects two nodes and is used to represent interdependency. Two connected nodes are called a pair. Nodes in a pair may be of the same type (homogenous) or different types (heterogeneous).

The semantics of any individual pairing is indicated by a textual description associated with the arc connecting the two nodes. The relationship is read according to the direction of the arrow. For instance, Figure 3 is read "LEAN is a type of Modelling Language".

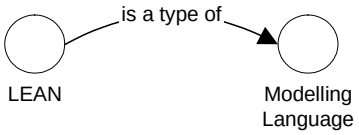

Fig. 3: A LEAN Relationship

The semantics that can be conveyed using LEAN pairings are unlimited. The types of relationships that will be needed to create an EA will vary depending on the particular enterprise, industry or user group.

6 The LEAN Ontology

The ontology is developed with the aim of specialising the metaphor so that it conveys the meaning required to describe an enterprise. Thus, the LEAN ontology is comprised of the following:

- A set of Universal nodes that are pre-specified (Agent, Action, Rule and Resource).
- A set of subtypes of each of the universal types. These are user-defined.
- A set of relationships between the LEAN nodes. These are also user-defined.

	LEAN NODE PAIRINGS								
	HOMOGENOUS PAIRINGS				HETEROGENOUS PAIRINGS				
RELATIONSHIP SET	Action Action	Agent Agent	Resource Resource	Rule Rule	Action Agent	Action Resource	Action Rule	Agent Rule	Resource Rule
is a type of	✓	✓	✓	✓	×	×	×	×	×
supports	×	×	✓	×	×	×	×	×	×
interfaces with	×	×	✓	×	×	×	×	×	×
is a part of	✓	✓	✓	✓	×	×	×	×	×
precedes	✓	×	×	×	×	×	×	×	×
reports to	×	✓	×	×	×	×	×	×	×
performed by	×	×	×	×	✓	×	×	×	×
uses	×	×	×	×	×	✓	×	×	×
produces	×	×	×	×	×	✓	×	×	×
complies with	×	×	×	×	×	×	✓	✓	✓
has applicable	×	×	×	×	×	×	✓	✓	✓
supports goal	×	×	×	×	×	×	✓	×	×

✓ = relationship allowed. × = relationship not allowed.

Table 1: Mapping between a generic relationship set and the range of possible node pairings

There are several significant implications that arise from the fact that LEAN users can define their own subtypes and relationships.

In terms of the subtypes that a user can create, we know the classes to which each of these subtypes will belong, as they can only be subtypes of one of the four Universal types defined in LEAN.

In terms of the relationships that a user can create, there is more flexibility. In fact, any number of relationship types may be defined. We term a collection of relationships a 'Relationship Set'. A unique relationship set could be created for every EA project. However, since it is likely that certain 'generic' relationships are likely to be used frequently, even in different environments, these are provided as part of the LEAN syntax. These generic relationships are referred to as 'Reference Relationships'. Table 1 shows these Reference Relationships and indicates to which node pairings each type of relationship applies. In effect, this generic LEAN relationship set serves as a starting point for developing an enterprise-specific relationship set.

It will be noted that relationships in the Relationship Set falls into two sets: those between homogenous pairs of nodes and those between heterogeneous pairs of nodes. Note also that the semantics of the heterogeneous relationships are such that the arrow on the arc will always point away from the Action type. This custom makes it easier to remember how these graphs are drawn and read.

7 Examples

Figure 4 shows a graph that uses subtypes or instances of each of the universal types. This graph depicts the following scenario: a salesperson develops a sales campaign using the CRM system in accord with privacy regulations.

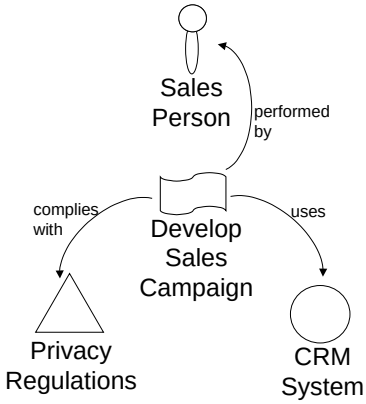

Figure 4 – A simple LEAN graph

Figure 5 shows a more complex graph that represents the student administration systems at a university. This model was produced as part of a commercial project to develop a university enterprise architecture using LEAN.

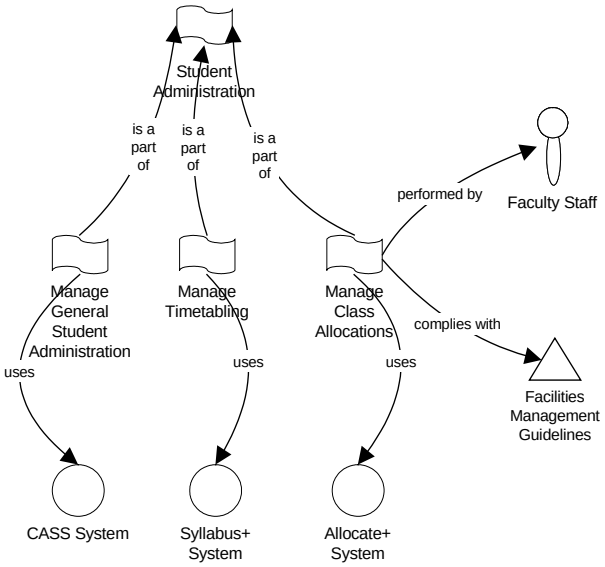

Figure 5 - A Student Administration LEAN Graph

8 Conclusion

Specialised languages have been developed for the modelling of specific ICT domains such as application, data, infrastructure or network architectures. However, few of these languages supports the creation of high-level, conceptual systems models that extend across a range of ICT domains. Of the languages that do meet these criteria, none have yet garnered broad support within the EA community.

While there have been several attempts to solve this problem, none can be said to be highly successful. The approach presented in this paper is quite novel in that it combines elements of cognition, linguistics, social theory and technology, to provide a methodology for identifying and developing conceptual metaphors into unified EA languages.

The use of a unified language that is based on a single highly conceptual metaphor will provide several benefits over models that are produced using multiple languages (Khoury and Simoff, 2005)

Further research into the methodology presented in this paper, and on the LEAN language itself, continues. This research is taking place both within the general EA practitioner community and also within a commercial IT organisation. Results from these studies will be used to refine both the language and the LEAN modelling tool.

It should be noted that one of the primary attributes required by any successful EA modelling language is that it must be simple and easy to use (Rostad, 2000)

(Solberg, 2000). LEAN's lack of complexity makes it suitable for use by the business community as well as IT specialists and academics. The graphical representations have been deliberately designed to be easy to draw, making the language ideal for collaborative work such as EA design workshops.

As ICT environments continue to increase in complexity, the role of enterprise architectures becomes more vital. We believe that the methodology presented in this paper helps to overcome a serious shortcoming of contemporary EA modelling approaches by allowing the development of truly unified, EA modelling languages.

6 References

Giddens, A. (1984) *The Constitution of Society: Outline of the Theory of Structuration*, Polity Press, Cambridge.

Gruber, T. R. (1993) Toward Principles for the Design of Ontologies Used for Knowledge Sharing *International Journal Human-Computer Studies***,** 907-928.

Henderson-Sellers, B. and Bulthuis, A. (1998) *Object-Oriented Metamethods*, Springer-Verlag, New York.

Hirst, G. (2003) In *Handbook on Ontologies in Information Systems*(Eds, Staab, S. and Studer, R.) Springer, Berlin, pp. 209-230.

Khoury, G. R. and Simoff, S. J. (2003) Elastic Metaphors: Expanding the Philosophy of Interface Design *Conferences in Research and Practice in Information Technology: Computers and Philosophy 2003,* **37**.

Khoury, G. R. and Simoff, S. J. 2005 Philosophical Foundations for a Unified Enterprise Modelling Language *Computers and Philosophy CAP2005 - to be held in 2005* Bangkok, Thailand

Maier, M. W. and Rechtin, E. (2000) *The Art of Systems Architecting*, CRC Press, Boca Raton.

Noran, O. (2003) An analysis of the Zachman framework for enterprise architecture from the GERAM perspective *Annual Reviews in Control,* **27,** 163-183.

Polovina, S. (1993) Loughborough University of Technology.

Rostad, C. C. (2000) In *Enterprise Modeling: Improving Global Industrial Competitiveness*(Eds, Rostadas, A. and Andersen, B.) Kluwer Academic Publishers, Boston, pp. 119-136.

Solberg, C. (2000) In *Enterprise Modeling: Improving Global Industrial Competitiveness*(Eds, Rostadas, A. and Andersen, B.) Kluwer Academic Publishers, Boston, pp. 183 -199.

Way, E. C. (1991) *Knowledge Representation and Metaphor*, Kluwer Academic Publishers, Dordrecht.